# Integration of Object Recognition, Color Classification, and QR Decoding for the Purposes of an Intelligent Mobile Robot

RADOSLAV VASILEV, NAYDEN CHIVAROV, VALENTINA IVANOVA
Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences,
1000 Sofia,
BULGARIA

*Abstract:* - The present work is part of the development process of a distributed platform for an intelligent mobile robot, with the proposed vision module intended for future integration into the overall architecture. Thanks to its built-in simulation testing capabilities, the functionality of the module can be validated even in the absence of a fully constructed physical robot. The vision module is part of the platform's intelligent core, which conceptually and functionally unifies algorithms, modules, and interfaces that collectively enable environmental perception, information processing, and decision-making by the robot. The vision module integrates algorithms for object recognition, color classification, and QR code decoding, executing them in a logical sequence to ensure efficient processing of visual information. Furthermore, it serves as an entry point to a broader logical model within the intelligent core. The paper presents the role of the vision module in the platform, its connection with other modules, and the conceptual guidelines for future development.

*Key-Words:* - Object recognition, OpenCV, Color classification, QR decoding, Intelligent mobile robots, Dynamic environment, Perceptual anchoring, Artificial intelligence.

Received: April 23, 2024. Revised: February 6, 2025. Accepted: March 9, 2025. Published: May 16, 2025.

## 1 Introduction

Intelligent mobile robots (IMRs) are agents physically embedded in the real environment through sensors and actuators that enable them to perceive, analyze, and interact with the surrounding world. The behavior of IMR is inherently linked to their environment and assigned tasks, which necessitates the development of flexible software-hardware architectures that integrate perception, logical processing, and control. With the growing demand for IMR in areas such as logistics, industry, and service operations, there is an increasing need for platforms capable of processing sensor data in real time and making autonomous decisions in dynamic environments, [1].

The present work is part of the process of developing a distributed platform for IMR. The platform includes distributed modules for perception, logical processing, and control, which are tested step by step in both simulated and real environments. One of the core components of the platform is the vision module, which integrates algorithms for object recognition, color classification, and QR code decoding. The module processes images from real cameras and simulated scenes, implementing a sequential logic for analyzing visual information.

The vision module is connected to a broader logical model, in which the results of perception will be used for decision-making based on context and semantic dependencies, through the process of perceptual anchoring, [2]. Therefore, it is part of the platform's intelligent core – a functional integration of modules and algorithms that ensures adaptability and autonomy in the behavior of the IMR.

At this stage, the functionality of the vision module is validated through tests with simulated and real images, allowing early evaluation without a fully completed physical robot. The focus is on building an integrated and extensible platform that can be upgraded with additional algorithms and sensors.

## 2 Architecture of the Distributed Platform

Different types of distributed platforms for IMR have been a subject of interest and research for years, [3], [4]. The architecture of our distributed platform for IMR represents the interaction between the robotic platform and the computer station (Figure 1), with emphasis placed on the integration

of modules responsible for perception, logical processing, and decision-making.

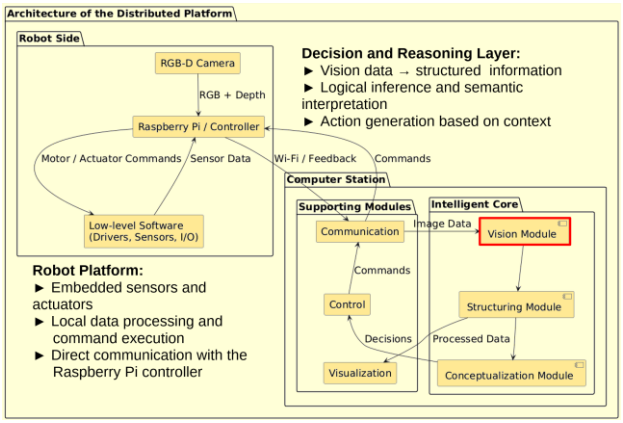

Fig. 1: Distributed platform for IMR

The computer station implements the vision module, which serves as the foundation for building the intelligent core of the platform. The purpose of the intelligent core is to integrate three main modules, each of which performs a specific role in the processes of perception, understanding, and decision-making:

- *Vision module* – Responsible for the perception and initial processing of visual (perceptual) information. This module extracts relevant features from images captured by a camera or other visual sources. It includes functionalities such as object recognition, color classification, and decoding of visual markers (QR codes). Additionally, it provides spatial localization of the recognized objects using a depth (D) camera. The results from the vision module serve as input data for the next module, which is responsible for structuring the information.
- *Structuring module* – Organizes and transforms the data received from the vision module (or other sensors) into structured representations. This is where the necessary structures and classes are built to logically represent the perceptual information. Static links between objects are created, and dependencies, hierarchies, and spatial relationships are defined. The objects in the system will be instances of these structured descriptions. The goal is to prepare the information for deeper conceptual interpretation.
- *Conceptualization module* – Performs semantic interpretation of the structured information. By using logical rules, symbolic representations, and knowledge bases, this module creates an internal model of the environment that can be used for reasoning, planning, and decision-making. This is where the transition occurs from "*what is seen*" to "*what it means*" and "*what should be done*".

The development of the intelligent core with these three main modules will enable the IMR to perceive the world not only as a set of pixels or objects but as a logically connected and understandable environment in which informed and autonomous decisions can be made.

The robotic side of the platform includes an embedded controller (Raspberry Pi) connected to an RGB-D camera and low-level software that manages the sensors and actuators of the IMR. This part of the architecture is responsible for collecting sensor data, performing initial processing, and executing commands received via Wi-Fi from the controlling computer station.

Figure 2 shows the actual prototype of the IMR used in the current phase of development and testing.

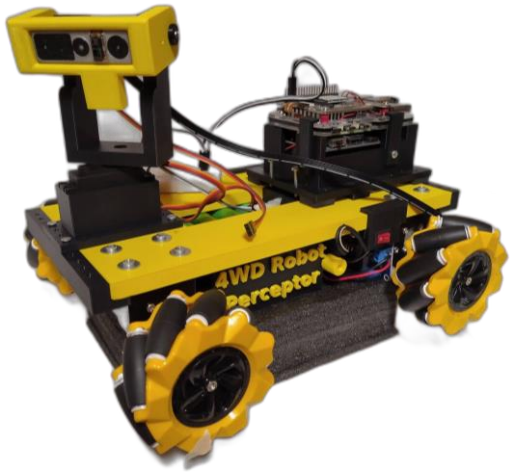

Fig. 2: Prototype of the IMR

# 3 Vision Module

## 3.1 Overview

The vision module, which is part of the intelligent core, integrates the following algorithms (Figure 3):

- *Object recognition*: Identifying and localizing different types of objects in the input images.
- *Color classification*: Analyzing the colors of the recognized objects in order to determine their color within one of six possible color categories.
- *QR decoding*: Extracting information from QR codes located within the recognized objects.
- *Depth measuring*: Calculating the distance to recognized objects using data from a depth (D) camera.

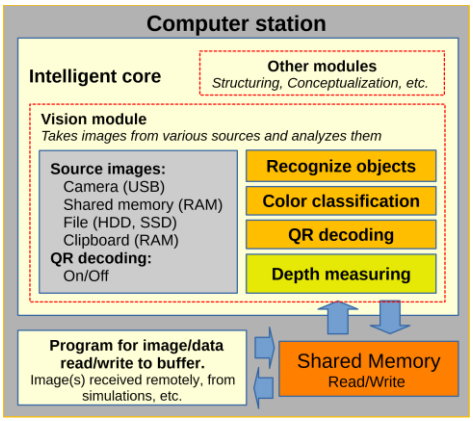

Fig. 3: Intelligent core block with vision module

For the development of the algorithms in the vision module, system libraries for working shared memory and the OpenCV library are used, providing a comprehensive solution for image processing and visualization [5], [6].

Figure 4 (Appendix) shows the block diagram of the program within the vision module. The components shown in the block diagram have the following meaning:

- *Initialization* – Creating variables, image display windows, a trackbar (GUI slider) window for setting RGB thresholds (used for filtering the input image), and initializing the camera.
- *Source selection menu* – At this stage, for experimental purposes, the program provides an interactive menu for selecting the image source (Figure 5).

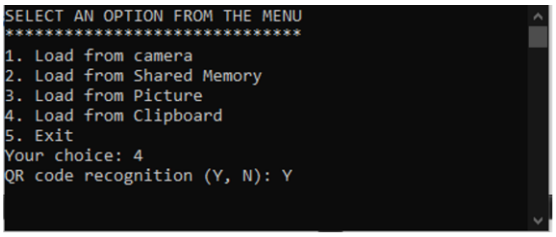

Fig. 5: Program menu

Table 1 describes the different selection options.

Table 1. Menu with selection options

| № | Menu option | Description |
|---|---|---|
| 1 | **Load from Camera** | Real-time image acquisition from a USB camera. |
| 2 | **Load from Shared Memory** | Loading images from shared RAM memory used by other programs or cameras that write data into memory. |
| 3 | **Load from Picture** | Opening an image file from the local disk. |
| 4 | **Load from Clipboard** | Loading an image saved in the system clipboard (e.g., from an external application). |
| 5 | **Exit** | Exit the program. |

- *Cyclic image processing* – The cycle ensures continuous updating of the images. Each image undergoes initial processing. This initial processing is performed using a trackbar, which allows the user to set lower and upper thresholds for the pixel values of the colors, thereby reducing noise in the image (Figure 6).

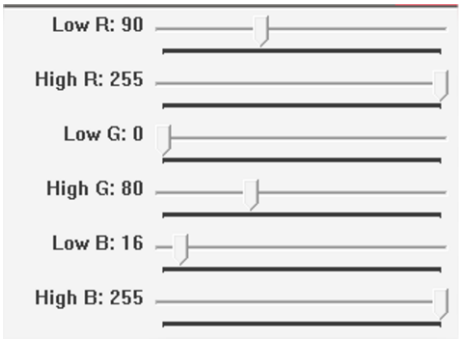

Fig. 6: Trackbar for setting pixel values of colors

After the initial image processing, the images are passed to an object recognition algorithm, followed by a color classification algorithm, and finally QR decoding.

- *Program termination* – The program ends its execution if the user selects the exit option from the menu or presses the “e” or “E” key on the keyboard.

At this stage, the main goal of the development is to create a functional and extensible foundation upon which the intelligent core of the platform can be built. The integration of object recognition, color classification, and QR decoding algorithms follows a logically and functionally justified sequence, providing a stable basis for the future development of the platform.

## 3.2 Object Recognition

Object recognition is performed on predefined classes. For the purposes of the experiments, simple 2D object classes (shapes) with potential applications in robotics have been selected. The recognized objects include: circles, triangles, rectangles, pentagons, and hexagons. The algorithm is robust against rotation and scaling, making it suitable for real-world conditions with dynamically positioned objects.

At this stage, storing information about the objects is not supported. During recognition, the system determines the class of a given object and its characteristics while it is present in the currently processed frames. Each detected object is classified, visualized, and labelled with an appropriate tag and unique number.

In summary, the recognition algorithm goes through the following steps:

1. *Selecting the image source* – The image can be loaded from a USB camera, file, system clipboard, or shared memory (Table 1).
2. *Threshold-based color segmentation* – The color filtering is performed using the *inRange()* function based on the RGB threshold values set through the trackbar (Figure 6), creating a binary image. Pixels whose values fall within the specified color boundaries are marked as white (value 255), while all other pixels are marked as black (value 0). This isolates the color of the target objects in the image, thereby separating them from other elements. The HSV model is more resistant to lighting changes and is often the preferred model for tasks such as object recognition and color segmentation, as demonstrated in studies, [7], [8]. At this stage, in our implementation, we use the RGB model for its intuitiveness, easy processing, and good visualization.
3. *Contour detection* – Closed contours (regions) are extracted from the binary image using the *findContours()* function.
4. *Shape approximation* – Each contour is simplified using the *approxPolyDP()* function to determine the number of vertices and the corresponding object class.
5. *Object recognition by shape* – After approximating the contours, an analysis of the number of vertices in each contour is performed. Based on this number, the geometric class of the object is determined: 3 vertices – triangle; 4 vertices – rectangle; 5 vertices – pentagon; 6 vertices – hexagon; more vertices or contours approaching a circle – circle.

The final result of these steps is the recognition and classification of objects by shape, which are visualized in a window with corresponding labels describing their geometric class and identifier.

The recognition algorithm has specific features, which are expressed in the fact that, at this stage of development, no specific algorithms for compensation during changes in lighting and colors are used, limiting its robustness in dynamic conditions. The recognition is sensitive to:

- *Image quality* – affected by noise, low resolution, poor focus, unstable connection, or poor lighting;
- *Object placement* – shading, reflections, distance, and viewpoint affect visibility and scale;
- *Determining color thresholds* – performed manually using a trackbar, which creates difficulties when changes occur in the external environment, a characteristic of a real-world IMR environment.

Although various approaches exist for automatically adapting to changes in lighting and colors, such as locally calculating threshold values or using optimization algorithms for segmentation, these are not implemented in the current version, [9], [10]. Such compensatory algorithms are planned for the next stage of development of the vision module, when the focus will shift toward improving robustness in dynamic environments.

After the recognition algorithm, each object is passed for further color classification.

### 3.3 Color Classification

The color classification model is built in Matlab and is based on a generalized mathematical model for linear decision filters (linear classification), [11]. Based on this model, an algorithm in C++ has been developed, which is included in the vision module and classifies the recognized objects into one of the following 6 colors within an acceptable time: *Red, Green, Blue, Magenta, Cyan, Yellow*.

The construction of the model in Matlab follows these steps:

1. Building a 3D color model in the RGB space;
2. Converting the 3D color space to 2D;
3. Creating lines for 2D color space separation;
4. Testing the model.

Implementation of the steps:

1. *Building a 3D color model in the RGB space.*

The RGB color space is described in [12]. To build the 3D model, this space is discretized along the three axes in the range [0, 1] with a step size of 0.02, resulting in 51 values per axis (50 intervals) and a total of 132,651 points. The choice of 51 values was made to ensure the algorithm's performance-higher values significantly slow down execution without leading to substantial improvements in results. A triple loop was used to implement the discretization. The resulting points form a 3D color matrix that covers all possible combinations of RGB values at the selected resolution. The created model is stored in a table and serves as the basis for color classification.

2. *Converting the 3D color space to 2D.*

The next step is the normalization of the RGB values, which allows the colors to be represented in

a two-dimensional space. The normalized values are calculated using the following formulas:

$$r = R / (R + G + B), g = G / (R + G + B),$$
$$b = 1 – (r + g),$$

where the coordinates r and g define the position of each color in the two-dimensional plane. Figure 7 shows the two-dimensional *rgb* space, which is used for visualization and classification of colors.

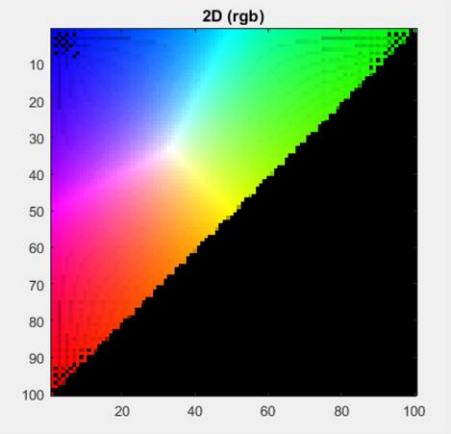


Fig. 7: Two-dimensional rgb space

For more efficient processing, the values of r and g are scaled from [0, 1] to [1, 100], and the resulting integer values are used as indices in a two-dimensional table (array). This enables fast visualization and classification of colors. The first 6 indices obtained from the formulas are shown in Figure 8.

The black pixels in the figure are the result of rounding the real values of the r and g coordinates (in the range [0, 1]) to integer values for faster indexing in the two-dimensional space. This does not affect the classification algorithm, as the graph in Figure 7 is for demonstration purposes and is not directly used in the calculations. The classification itself is based on a generalized model that uses lines to separate the color regions. In this way, even colors falling within the black pixels of the visualization are classified correctly.

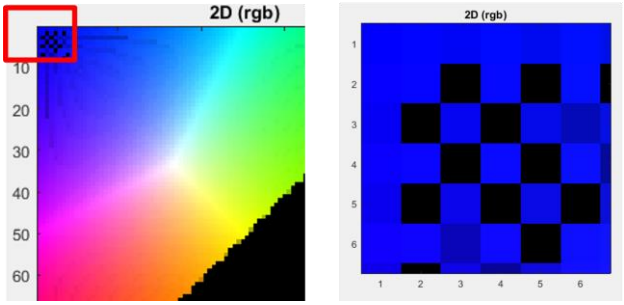


Fig. 8: Indexing of *r* and *g* coordinates

*3. Creating lines for 2D color space separation.*

As seen in Figure 9, several primary colors stand out in the two-dimensional space: *Red, Green,* and *Blue*. There are also three more colors that are well-defined in the graph: *Magenta, Cyan,* and *Yellow*. Since these colors are clearly defined, the classification model has been built for these 6 colors. To achieve color classification of the objects, lines have been drawn to separate these color regions.

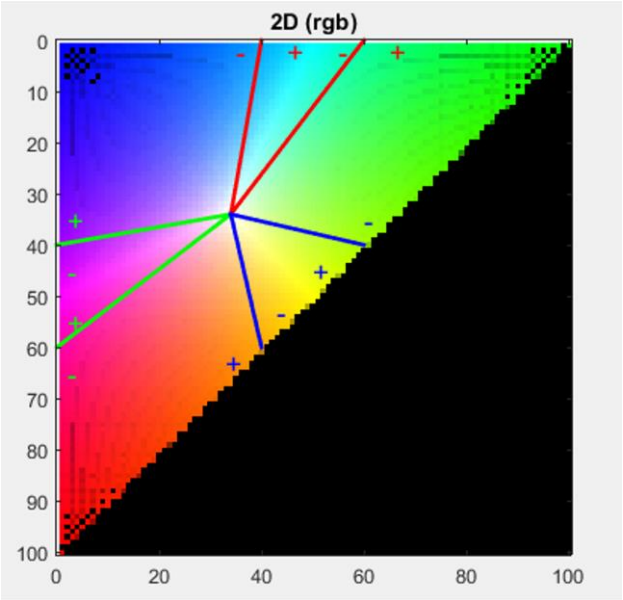


Fig. 9: Lines in two-dimensional rgb space

The lines are mathematically described by linear equations. Although in the figure they are shown as finite segments, in the mathematical model they are considered to be infinite in both directions.

The coefficients for the 6 lines of the model have the following values:

*Dbr1 = [ -6 -34 1360] % first green line*
*Dbr2 = [-26 -34 2040] % second green line*
*Dbg1 = [34 6 -1360] % first red line*
*Dbg2 = [34 26 -2040] % second red line*
*Drg1 = [-26 6 680] % first blue line*
*Drg2 = [-6 26 -680] % second blue line*

The signs (+) and (-) placed next to the lines determine the values of *Zbr1*, *Zbr2*, *Zbg1*, *Zbg2*, *Zrg1*, *Zrg2*, which are calculated in the procedure shown below.

Color classification procedure:

- The average color value for the recognized object in the 3D RGB space is determined, with values in the range [0, 255].

- The obtained RGB color is converted into the 2D *rgb* space in the range [0, 1]. The values for *rg* are used to determine the following two values:
  *x_green* = 1 + g * 99*;*
  *y_red* = 1 + *r* * 99*;*
  where: *x_green* and *y_red* are the color values of the recognized object in the two-dimensional space.
- The obtained values for *x_green* and *y_red* are substituted into the equations of *Zbr1*, *Zbr2*, *Zbg1*, *Zbg2*, *Zrg1*, *Zrg2* shown below, and only the resulting sign is considered.
  *W = [x_green; y_red;* 1*];*
  *Zbr1 = Dbr1 * W;*
  *Zbr2 = Dbr2 * W;*
  *Zbg1 = Dbg1 * W;*
  *Zbg2 = Dbg2 * W;*
  *Zrg1 = Drg1 * W;*
  *Zrg2 = Drg2 * W;*
  *IF (Zbr1* > 0 *&& Zbg1* < 0*) disp('blue');*
  *IF (Zbg1* >= 0 *&& Zbg2* <= 0*) disp('cyan');*
  *IF (Zbg2* > 0 *&& Zrg2* < 0*) disp('green');*
  *IF (Zrg2* >= 0 *&& Zrg1* <= 0*) disp('yellow');*
  *IF (Zrg1* > 0 *&& Zbr2* < 0*) disp('red');*
  *IF (Zbr2* >= 0 *&& Zbr1* <= 0*) disp('magenta');*

In the vision module, color classification of the recognized object is performed in the *colorClassification()* function. This function implements the program code of the model from Matlab, which classifies the object's color.

4. *Testing the model.*
The goal of testing the Matlab model is to verify whether the algorithm correctly classifies the specified colors. For this purpose, manually averaged RGB component values of objects with a known color are entered.

Parameters and results from two of the many tests conducted:

*Test 1:*
*Object color (RGB): R = 255, G = 0, B = 0*
*Command in Matlab: >> color_classification_objects*
*Classification result: red*
*Color location in the 2D color space: a green star*

*Test 2:*
*Object color (RGB): R = 255, G = 255, B = 0*
*Command in Matlab: >> color_classification_objects*
*Classification result: yellow*
*Color location in the 2D color space: a grey star*

Figure 10 shows the position of each object's color in the two-dimensional color space as a result of the classification of a red and a yellow object.

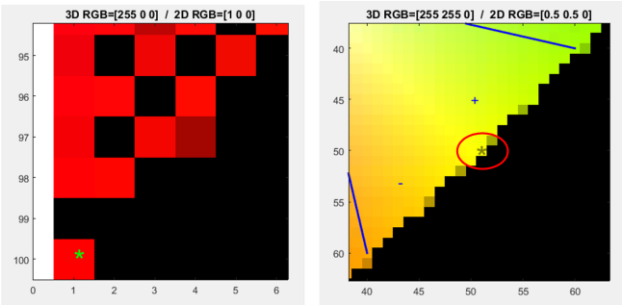


Fig. 10: Classification results for a red and a yellow object in the two-dimensional color space

The classified colors and their placement in the two-dimensional color space match the expectations, which confirms the validity of the model.

### 3.4 QR Decoding

QR codes (Quick Response codes) are two-dimensional matrix barcodes designed to store enough data for practical applications, while enabling fast scanning, omnidirectional readability, and error correction, [13]. In intelligent robotics, they are often used for object identification, navigation, and localization, [14], [15], [16], [17].

The vision module has implemented functionality for recognizing and decoding QR codes, taking into account specific features aimed at enhancing the reliability and robustness of the decoding process.

Working with QR codes involves the following main steps:

1. *Activating the decoding* – The functionality is activated by selecting from the program interface (Figure 5).
2. *Detecting the code* – The program locates the QR code within the input image, limited to the recognized object.
3. *Extracting information* – If a valid code is found, it is decoded and the result is displayed as a text value.

In the current implementation, the QR code must be located within the already recognized object. This condition serves as a deliberate protection against noise, visual artifacts, or the presence of codes that do not logically relate to the scene. This ensures that the decoded QR codes have contextual meaning and are associated with real objects identified by the system.

Extended implementation variants, based on fiducial markers such as *AprilTag*, are not included at this stage but are planned for future development stages of the architecture, [18], [19], [20].

Within the vision module, the QR decoding functionality is implemented through the *QRDetector()* function, which takes an image of a recognized object and outputs the results to the console.

*QRDetector* uses the *QRCodeDetector* class from the *objdetect.hpp* header file of *OpenCV*. The method *detectAndDecode()* performs the following actions:

- Detecting the QR code in the image (if present);
- Decoding the content;
- Returning the following information:
  - *data* – the decoded text;
  - *bbox* – coordinates of the detected bounding box;
  - *rectifiedImage* – the rectified (restored) image of the QR code, which can easily be visualized if needed.

The results of the decoding (the text content, coordinates, and the restored image) will be stored in the structural module of the intelligent core (Figure 1) and will be accessible at a higher level of information processing.

# 4 Testing the Vision Module

## 4.1 Functionality

During testing, the following options were selected from the program's user menu (Figure 5):

- *Option 2: Load from Shared Memory* – For this experiment, the ESP32-CAM camera was used. The pentagon model was created in Blender and displayed on the monitor screen. The camera captures the screen and saves the images in the shared memory area, from where the vision module loads them. An additional program was developed to copy the obtained images into the shared memory. This program is part of the supporting modules block (Figure 1).
- *Option 4: Load from Clipboard* – The pentagon model created in Blender is copied directly to the system clipboard. The vision module loads the image directly from this area.

The results of the program execution are shown in Figure 11 (Appendix). In the left column, the result of analyzing an image obtained from the ESP32-CAM camera is displayed. The object is recognized as a pentagon (*PEN – pentagon*), the color is classified as blue, and the decoded QR code contains the text "*Chocolate products*".

In the right column, the result of analyzing the same pentagon is presented, but this time the image is loaded from the system *clipboard*. It is evident that the result of the second analysis is more accurate, as there is no noise in the image and the color is correctly recognized.

In the center of the object, the label *PEN 1* is visible, which stands for pentagon, and the number *1* is the identification number of the object, automatically assigned by the program. In the top-left corner, the defined color "cyan" is displayed. In the bottom-left corner, the values for the color in rg (Red-Green) are shown in a two-dimensional color scheme. At the bottom of the figure, information about the object and the QR decoding result is displayed.

For IMR, the use of real-time cameras is of interest, but due to the presence of noise in the images, the research with them is complicated. This necessitates the application of additional approaches for investigation and testing. For this reason, the program provides options for loading images from various sources. Based on the results obtained from the experiments, cameras that are most suitable for the IMR being used can be selected. For the prototype we are developing, we are using the *Orbbec Petrel A RGB-D camera*, [21].

Figure 12 (Appendix) illustrates additional examples with geometric shapes for which all stages of processing – object recognition, color classification, and QR code decoding – have been successfully implemented.

## 4.2 Practical Application

The architecture of the intelligent core is shown in Figure 13 (Appendix).

The vision module has been successfully integrated with the RGB-D camera, [22]. In the same source, the definition of "*program model*" is used, reflecting the initial concept of the software architecture – without a clear distinction between individual modules within the intelligent core. According to the new architectural concept, functionality is divided into separate modules that interact dynamically. Information from the RGB sensor of the camera is used for object recognition in the environment and for determining their characteristics, while the D-sensor is used to measure the distance to objects and calculate their spatial positioning relative to the camera. The obtained data can be used for controlling the IMR

based on the objects' placement, avoiding obstacles, optimizing routes, and performing other navigation tasks.

The recognition of two-dimensional objects and QR codes, which is one of the capabilities of the vision module, finds application in:

- *Navigation and routing* – using 2D markers for guiding along pre-defined paths;
- *Positioning* – using shapes or QR codes for precise positioning (e.g., in warehouse environments);
- *Light signals* – through color recognition in industrial environments;
- *Object sorting* – using QR codes and color classification in logistics.
- *Visual inspection* – using 2D shapes and geometric features for defect detection and classification in technical objects, [23].

The module's ability to process images from both real cameras and simulated 2D/3D scenes allows for preliminary testing, adaptation, and verification of algorithms in various simulated scenarios. This advantage is particularly valuable for identifying potential issues and optimizing system behavior without risking damage to the hardware.

To enhance robustness, the vision module employs the following mechanisms:

- Limiting recognition to predefined object classes;
- Color classification through the average RGB value of all object pixels;
- QR codes are recognized only within already identified objects, which eliminates the decoding of irrelevant information.

# 5 Development Opportunities for the Architecture

## 5.1 Activity and Expansion in the Architecture

The legend in Figure 13 (Appendix) shows the activity in the architecture. The intelligent core has three main modules, with the vision module currently being developed. The "*Structuring Information*" and "*Perceptual Anchoring and Conceptualization*" modules are actively under development. The meaning of the blocks in the intelligent core is recorded in Table 2.

Table 2. Main modules in the intelligent core

| Module | Function |
|---|---|
| **State Measurement** | Processes data from the IMU to determine the current state of the IMR (position, movement, orientation). |
| **Distance Measurement** | Analyzes information from LiDAR and ultrasonic sensors for obstacle detection and avoidance. |
| **Object Recognition, Color Classification, and QR Decoding. Spatial positioning (D-Camera)** | Uses an RGB-D camera for object recognition, color classification, and QR code decoding. Determines the spatial positioning of objects relative to the camera. |
| **Environment Measurement** | Works with data from temperature, humidity, and light sensors to assess the environment. |
| **Structuring Information** | A central module that integrates and structures all incoming data to make it compatible with logical processing. This is perceptual information that serves as a transition to intelligent behavior. |
| **Perceptual Anchoring and Conceptualization** | A logical module that transforms structured information into logical conclusions and action decisions, [24]. |

The structuring of perceptual information and the application of the perceptual anchoring process allow the IMR not only to recognize objects (percepts) but also to link them with corresponding symbols – semantic representations, which are used at the logical level. This connection between perception and symbolic representation creates the foundation for the process of conceptualization, transitioning from sensory information to internal knowledge that can be used for logical reasoning and decision-making.

The "*Other Library*" block, connected with a dashed arrow to the intelligent core in Figure 13, provides the possibility for integration with various libraries and algorithms. Table 3 shows examples of integration.

Table 3. Integration possibilities with libraries/algorithms

| Library/Algorithm | Function and Application |
|---|---|
| **Dijkstra's Algorithm** | Optimal route planning in graph-based environments (e.g., navigation for obstacle avoidance). |
| **Kalman Filter** | Capabilities for calculating the IMR state (position, speed, orientation) based on noisy sensor data. |
| **AI Methods** | Classification, prediction, and decision-making based on accumulated experience and trained models, [25]. |

### 5.2 Integration Possibilities with Additional Sensors

At the top of Figure 13 (Appendix), the real world is represented, where the IMR perceives objects and the state of the environment through various sensors. The obtained information is structured and subjected to logical interpretation before being used to generate commands to the IMR. Thus, sensor data is integrated with perceptual structures extracted from the cameras, creating a more complete and detailed picture of the surrounding environment.

Adding various sensors can significantly expand the capabilities of the intelligent core. Temperature, humidity, and light sensors can contribute to a better understanding of the environment and allow for adaptation to specific conditions, such as high temperature, increased humidity, or low light, [26], [27], [28], [29]. For example, detecting a change in lighting can lead to adaptations in the algorithms that process the images.

### 5.3 Integration Possibilities with Optimization Algorithms

The developed vision module can be used in motion optimization algorithms, such as Dijkstra's algorithm, [30], [31].

Through the vision module, the IMR can:

- Automatically identify nodes in the navigation graph using QR codes.
- Recognize color markers and geometric shapes that indicate the position and accessibility of paths.
- Adapt its route based on the state of the environment.
- Avoid blocked or occupied nodes by dynamically recalculating their paths.

Figure 14 (Appendix) shows the state of a simulated node after applying the current capabilities of the vision module.

The figure shows that all node markers have been successfully recognized regardless of their location and rotation.

In a real environment, markers can be created in various ways:

- Paint on the floor for fixed signs and paths.
- Colorful stickers or labels.
- RGB LED matrices that dynamically change their meaning depending on the state of the node.

Using the vision module, the IMR can perform the following tasks:

1. *Initialization* – Identifies its starting position by recognizing a QR code at the center of the node.
2. *Recognizing possible exits* – Cyan rectangles indicate allowable paths, and the QR codes within them provide information about the edges and weights in the navigation graph.
3. *Determining the direction of movement* – Triangles indicate the direction to the exit path, and their colors, when using RGB LED matrices, can be used for signaling:
   - *Red* – The node is occupied and cannot be used.
   - *Yellow* – The node is expected to become available soon.
   - *Green* – The node is free and accessible.
4. *Movement to the next node* – After selecting a path, the IMR moves along the pre-calculated route.

Although it is hypothetical, this node demonstrates how the vision module can use visual markers for navigation, allowing the IMR to recognize available routes, adapt its movement, and optimize its path in a dynamic environment.

## 6 Conclusion

The presented vision module is a functionally active and validated component of the intelligent core of the distributed platform for the IMR. Its development provides a foundation for future upgrades with additional algorithms, filters, and logical mechanisms.

The module integrates three main algorithms – object recognition, color classification, and QR code decoding. Additionally, functionality for working with the D-camera has been implemented, enabling the determination of spatial coordinates. Thanks to its flexibility, the module is applicable in both stationary and IMR.

Experimental results confirm that the module provides reliable object recognition (within predefined classes), color recognition (using predefined lines in the color space), and QR code decoding (located within recognized objects). The algorithms work stably even in environments with a high concentration of visual markers. The combination of visual markers with depth camera information extends the IMR's capabilities, enabling more precise perception of the spatial environment and adaptive behavior in real-time.

The intelligent core of the platform is under development. Its capabilities can be expanded through:

- Adding new algorithms and libraries;
- Integration of additional sensors.

The future development of the intelligent core includes:

- Structuring and storing data for recognized objects, supporting perceptual anchoring and logical reasoning;
- Scene reconstruction through analysis of spatial dependencies, enabling the creation of an internal representation of the environment;
- Expanding algorithms for 2D object recognition and adding algorithms for 3D recognition;
- Increasing the number of lines for color classification and implementing alternative approaches for color evaluation;
- Adapting to variable lighting and automatically determining color thresholds to improve robustness in different environments.

*References:*

[1] Rubio, F.; Valero, F.; Llopis-Albert, C. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 2019, vol. 16, no. 2. https://doi.org/10.1177/1729881419839596.

[2] Coradeschi, S.; Saffiotti, A. Perceptual Anchoring of Symbols for Action. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, Seattle, WA, USA, 4 August 2001; *vol. 1*, pp. 407–412.

[3] Fernández, I.; Mazo, M.; Lázaro, J.L.; Pizarro, D.; Santiso, E.; Martín, P.; Losada, C. Guidance of a mobile robot using an array of static cameras located in the environment. *Autonomous Robots* 2007, vol. *23, no. 4*, pp. 305–324. https://doi.org/10.1007/s10514-007-9049-4.

[4] Hanel, M.L.; Kuhn, S.; Henrich, D.; Grüne, L.; Pannek, J. Optimal Camera Placement to Measure Distances Regarding Static and Dynamic Obstacles. *International Journal of Sensor Networks* 2012, vol. *12*, no. 1, pp. 25–36. https://doi.org/10.1504/IJSNET.2012.047713.

[5] Microsoft. Using File Mapping, [Online]. https://learn.microsoft.com/en-us/windows/win32/memory/using-file-mapping (Accessed Date: April 29, 2024).

[6] OpenCV: Open Source Computer Vision Library, [Online]. https://github.com/opencv/opencv (Accessed Date: April 29, 2024).

[7] Kang, H.-C.; Han, H.-N.; Bae, H.-C.; Kim, M.-G.; Son, J.-Y.; Kim, Y.-K. HSV Color-Space-Based Automated Object Localization for Robot Grasping without Prior Knowledge. *Applied Sciences*, 2021, vol. 11, no. 16, 7593. https://doi.org/10.3390/app11167593.

[8] Hema, D.; Kannan, S. Interactive Color Image Segmentation Using HSV Color Space. *Science and Technology Journal*, 2020, vol. 7, no. 1, pp. 37–41. https://doi.org/10.22232/stj.2019.07.01.05.

[9] Navon, E.; Miller, O.; Averbuch, A. Color image segmentation based on adaptive local thresholds. *Image and Vision Computing,* vol. 23, no. 1, January 2005, pp. 69-85. https://doi.org/10.1016/j.imavis.2004.05.011.

[10] Xing, Z. An Improved Emperor Penguin Optimization Based Multilevel Thresholding for Color Image Segmentation. *Knowledge-Based Systems* 2020, vol. *194*, 105570. https://doi.org/10.1016/j.knosys.2020.105570.

[11] Gochev, G. *Kompyutarno zrenie i nevronni mrezhi*; Technical University – Sofia: Sofia, Bulgaria, 1998; pp. 83–84, 125–143. (In Bulgarian, Textbook).

[12] Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*, 4th ed.; Pearson: New York, NY, USA, 2018; pp. 400–408, [Online]. https://www.cl72.org/090imagePLib/books/Gonzales,Woods-Digital.Image.Processing.4th.Edition.pdf (Accessed Date: April 29, 2024).

[13] Tiwari, S. An Introduction to QR Code Technology, *2016 International Conference on Information Technology (ICIT)*, Bhubaneswar, India, 22–24 December 2016; IEEE: 2016; pp. 39–44. https://doi.org/10.1109/ICIT.2016.021.

[14] Zhang, H.; Zhang, C.; Yang, W.; Chen, C.-Y. Localization and navigation using QR code for mobile robot in indoor environment. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China, 6–9 December 2015*; IEEE: 2015; pp. 2501–2506. https://doi.org/10.1109/ROBIO.2015.7419715

[15] Sneha, A.; Sai Lakshmi Teja, V.; Mishra, T.K.; Satya Chitra, K.N. QR Code Based Indoor Navigation System for Attender Robot. *EAI Endorsed Transactions on Internet of*

*Things*, vol. 6, no. 21, p. e3, Apr. 2020. https://doi.org/10.4108/eai.13-7-2018.165519.

[16] Bach, S.-H.; Khoi, P.-B.; Yi, S.-Y. Application of QR Code for Localization and Navigation of Indoor Mobile Robot. *IEEE Access*, vol. 11, pp. 28384–28390, 2023. https://doi.org/10.1109/ACCESS.2023.3250253.

[17] Aman, A.; Singh, A.; Raj, A.; Raj, S. An Efficient Bar/QR Code Recognition System for Consumer Service Applications. In *Proceedings of the 2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, Novi Sad, Serbia, 26-27 May 2020; IEEE: 2020; pp. 127–131. https://doi.org/10.1109/ZINC50678.2020.9161778.

[18] Wang, J.; Olson, E. AprilTag 2: Efficient and robust fiducial detection. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea (South), 09-14 October 2016; pp. 4193–4198. https://doi.org/10.1109/IROS.2016.7759617.

[19] Krogius, M.; Haggenmiller, A.; Olson, E. Flexible Layouts for Fiducial Tags. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 03-08 November 2019; IEEE: 2019; pp. 1898–1903. https://doi.org/10.1109/IROS40897.2019.8967787.

[20] Wang, X. *High Availability Mapping and Localization*. Ph.D. Dissertation, University of Michigan, 2019, [Online]. https://hdl.handle.net/2027.42/151428 (Accessed Date: April 29, 2024).

[21] Orbbec. *Petrel-A User Manual*, July 2023, [Online]. https://www.orbbec.com/staging/wp-content/uploads/2023/07/Petrel-A-En.pdf (Accessed Date: April 29, 2024).

[22] Vasilev, R.; Chivarov, N.; Staikova, M. Distributed 3D Camera Distance Measurement System for Intelligent Mobile Robots. In *2024 International Conference Automatics and Informatics (ICAI)*, Varna, Bulgaria, 10–12 October 2024; IEEE: 2025; pp. 233–239. https://doi.org/10.1109/ICAI63388.2024.10851561.

[23] Melnyk, R.; Viazovskyy, P. Detection of Defects in PCB Images by Numbering, Measurement of Chain Features and Machine Learning. *WSEAS Transactions on Circuits and Systems*, vol. 23, pp. 305–317, 2024. https://doi.org/10.37394/23201.2024.23.30.

[24] R. Vasilev and D. Dimitrov, Logical Modeling of Dynamic Environments for Intelligent Robots Based on the Perceptual Anchoring Process. In *Proceedings of the Technical University - Sofia*, vol. 63, no. 2, pp. 49–58, June 14–16, 2013, [Online]. https://proceedings.tu-sofia.bg/volumes/Proceedings_Volume_63_book_2_2013.pdf (Accessed Date: April 10, 2024).

[25] Bodapati, R. B.; Srinivas, R. S.; Ramana Rao, P. V. Artificial Neural Network-Based Hybrid Controller for Electric Vehicle Applications. *WSEAS Transactions on Circuits and Systems*, vol. 23, pp. 192–201, 2024. https://doi.org/10.37394/23201.2024.23.20.

[26] Bonci, A.; Cheng, P. D. C.; Indri, M.; Nabissi, G.; Sibona, F. Human-Robot Perception in Industrial Environments: A Survey. *Sensors* 2021, vol. 21, no. 5, p. 1571. https://doi.org/10.3390/s21051571.

[27] Zhmud, V. A.; Kondratiev, N. O.; Kuznetsov, K. A.; Trubin, V. G.; Dimitrov, L. V. Application of ultrasonic sensor for measuring distances in robotics. *Journal of Physics: Conference Series*, vol. 1015, no. 3, p. 032189, May 2018. https://doi.org/10.1088/1742-6596/1015/3/032189.

[28] Suh, Y. S. Laser Sensors for Displacement, Distance and Position. *Sensors* 2019, vol. 19, p. 1924. https://doi.org/10.3390/s19081924.

[29] Lee, C.; Song, H.; Choi, B. P.; Ho, Y.-S. 3D scene capturing using stereoscopic cameras and a time-of-flight camera. *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1370–1376, August 2011. https://doi.org/10.1109/TCE.2011.6018896.

[30] GeeksforGeeks. How to Find Shortest Paths from Source to All Vertices Using Dijkstra's Algorithm, [Online]. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/.

[31] Alshammrei, S.; Boubaker, S.; Kolsi, L. Improved Dijkstra Algorithm for Mobile Robot Path Planning and Obstacle Avoidance. *Computers, Materials & Continua*, vol. 72, no. 3, pp. 5939–5954, April 2022. https://doi.org/10.32604/cmc.2022.028165.

# APPENDIX

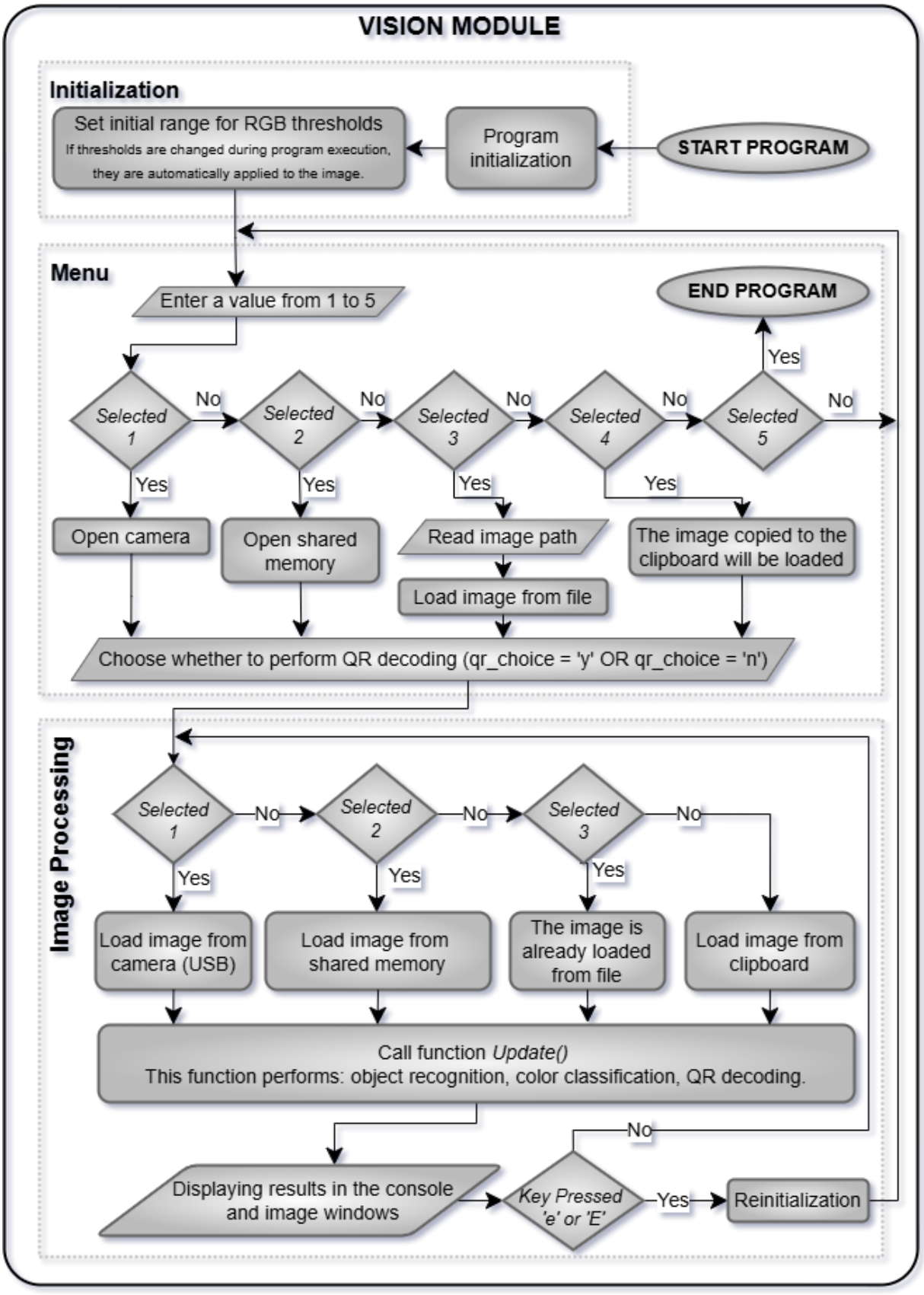


Fig. 4: Block diagram of the program in the vision module

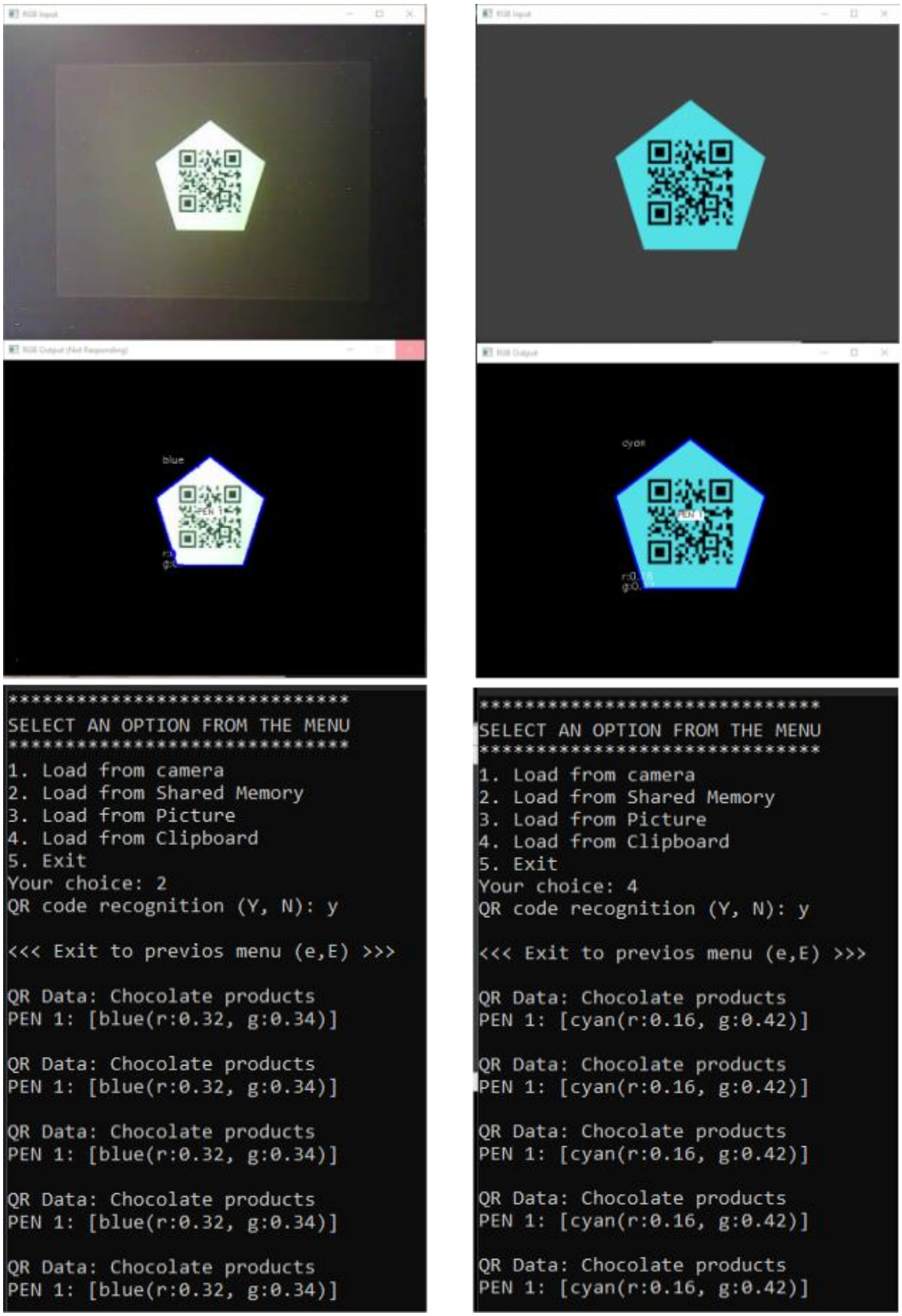


Fig. 11: Results after selecting Option 2 (left column) and Option 4 (right column) from the program menu

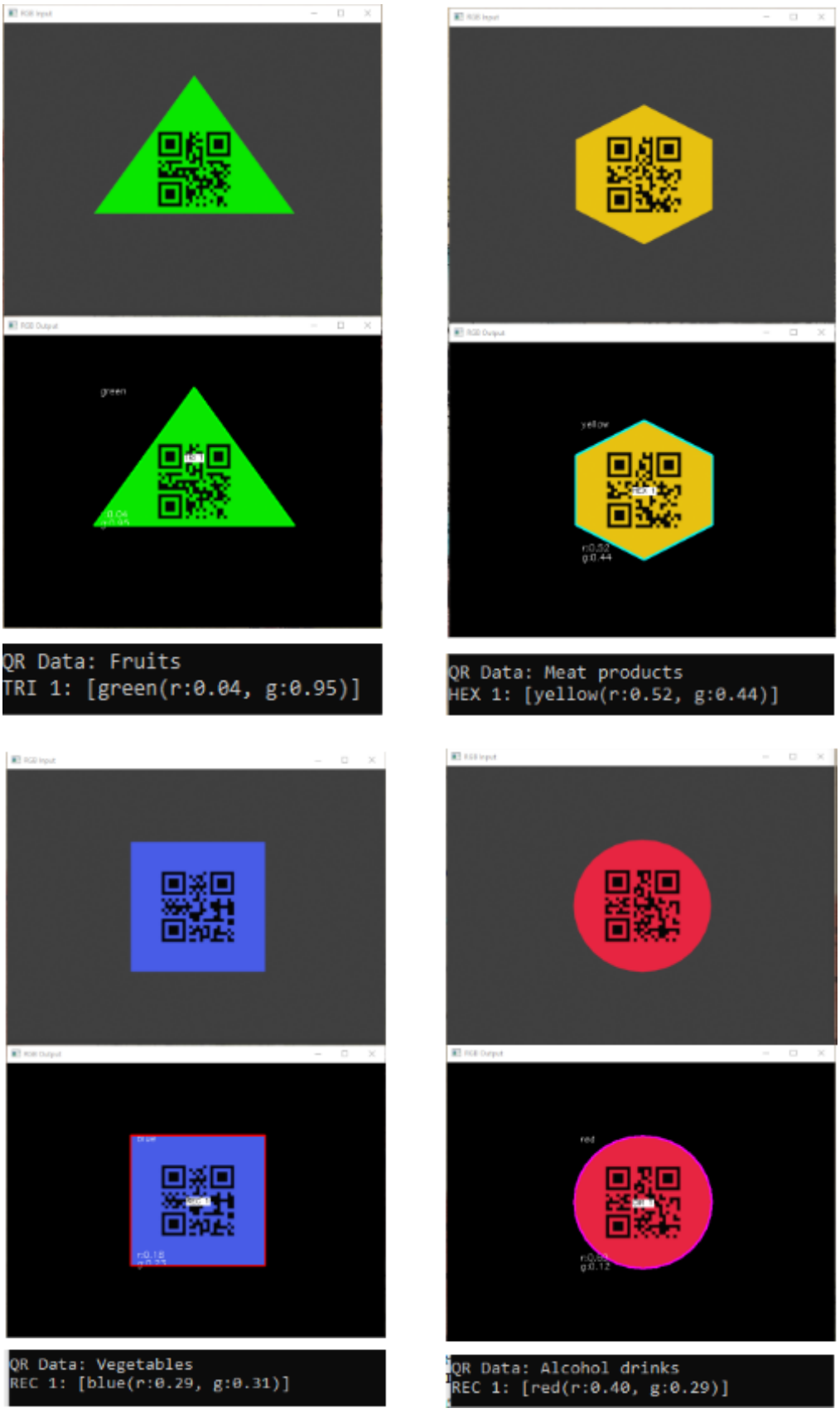


Fig. 12: Results from the vision module on different objects

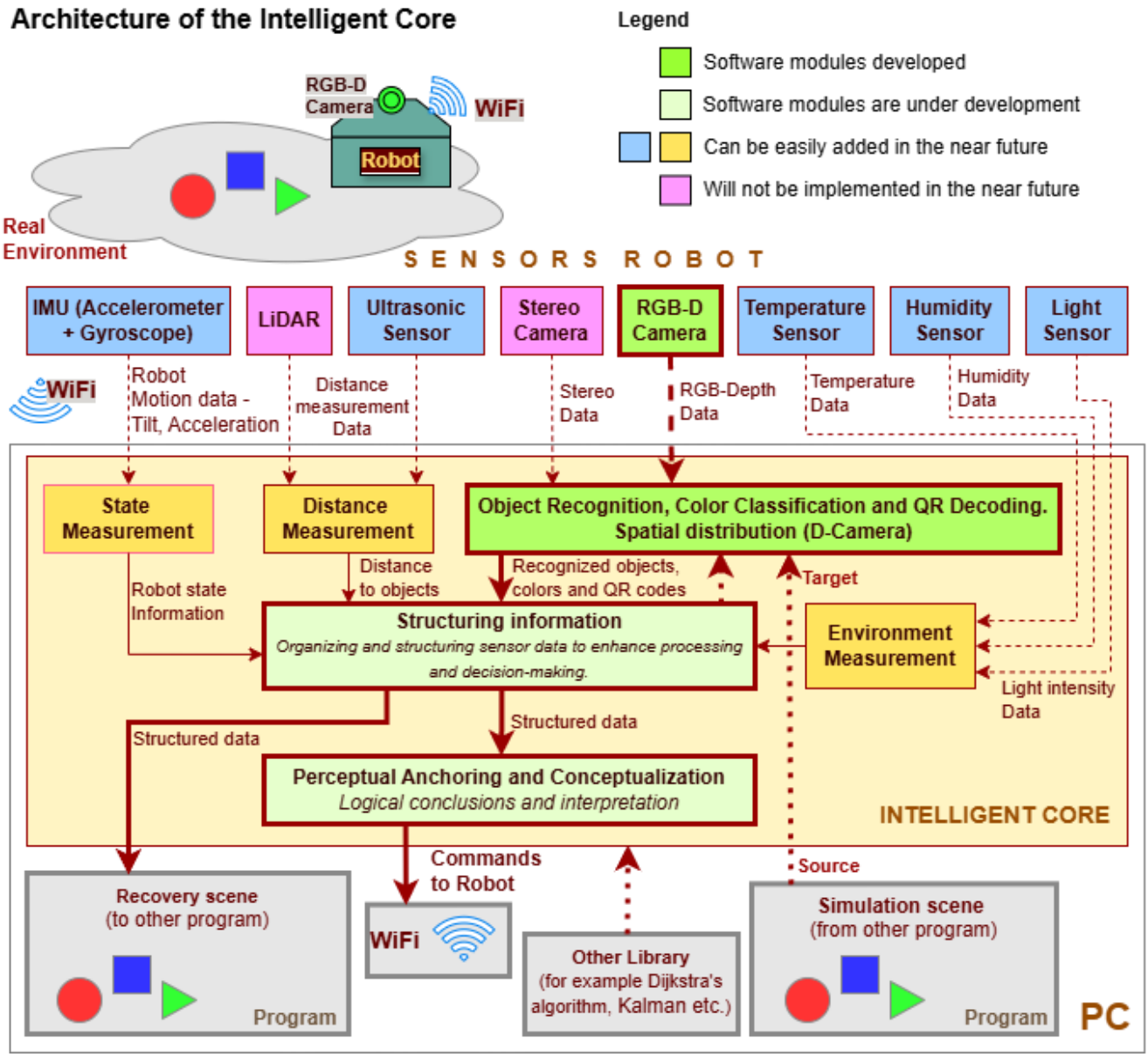

Fig. 13: Architecture of the intelligent core

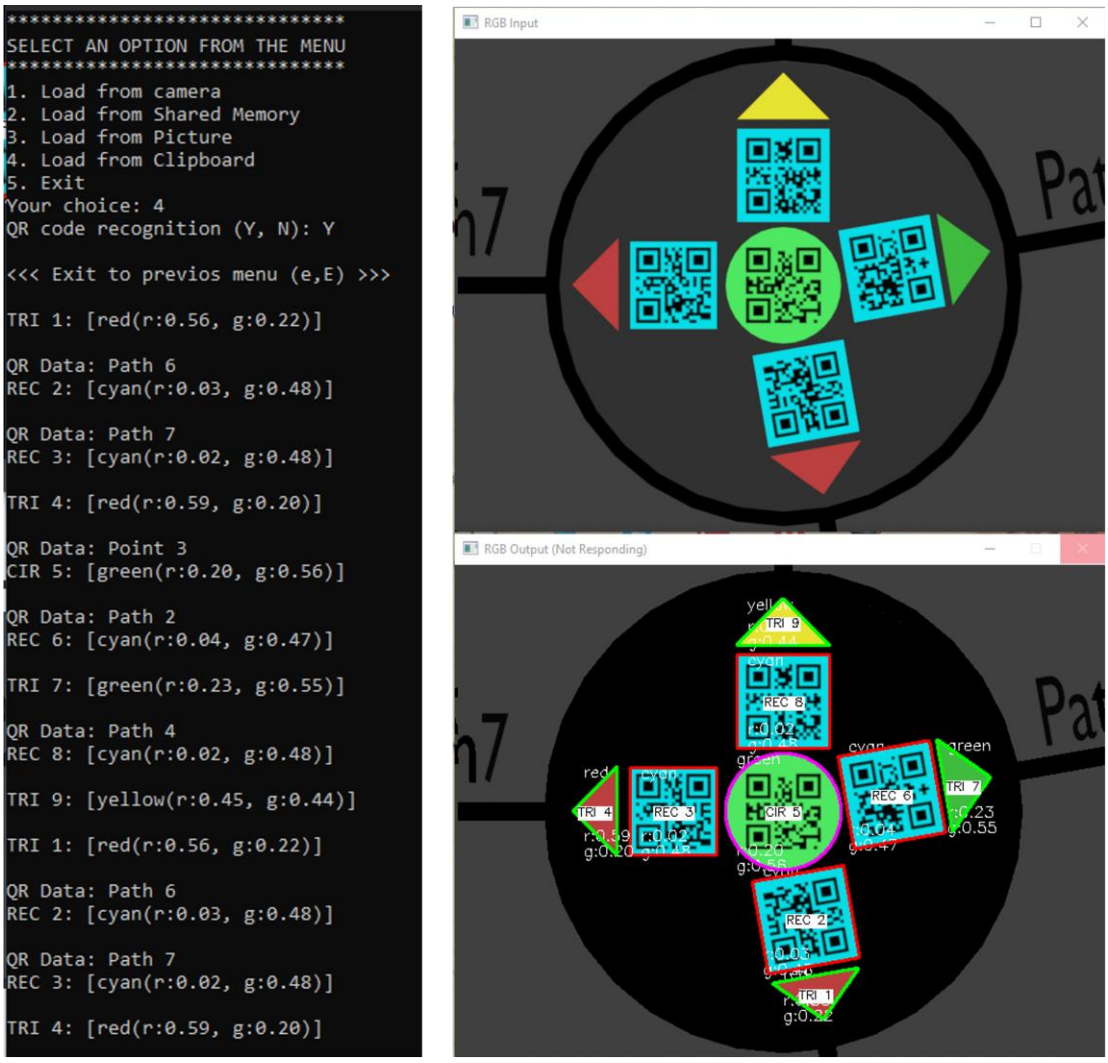

Fig. 14: Results after selecting Option 4 from the program menu with QR decoding enabled

**Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)**

- Radoslav Vasilev (R.V.): Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing-original draft preparation, writing-review and editing, visualization, project administration.
- Nayden Chivarov (N.C.): Conceptualization, supervision, validation, formal analysis, resources, software development, project administration, funding acquisition.
- Valentina Ivanova (V.I.): Methodology, validation, data curation, writing-review and editing.

All authors have read and agreed to the published version of the manuscript.

**Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself**

No funding was received for conducting this study.

**Conflict of Interest**

The authors have no conflicts of interest to declare.